強化学習をとり入れたボルツマン・マシンによる非線形計画問題の大域的最適解の解法 (数理最適化から見た「凸性の深み,非凸性の魅惑」)

AUTHOR(S):

小熊, 崇

CITATION:

小熊, 崇. 強化学習をとり入れたボルツマン・マシンによる非線形計画問題の大域的最適解の解法 (数理最適化から見た「凸性の深み,非凸性の魅惑」). 数理解析研究所講究録 2004, 1349: 7-22

ISSUE DATE:

2004-01

URL:

http://hdl.handle.net/2433/24857

数理解析研究所講究録 1349 巻 2004 年 7-22

強化学習をとり入れたボルツマン・マシンによる非線形計画問題の大域的最適解の解法

ソニー デジタル ネットワーク アプリケーションズ (株)　小熊 崇 (Takashi Oguma)
Sony Digital Network Applications Inc.

1. はじめに

非線形計画問題の解法は従来さまざまな研究がなされ、いくつもの手法が提案されている。しかしながら、いずれもごく特別な形の問題に対する解法が見出されているに過ぎず、線形計画法における単体法のような統一した解法は存在しない。その最大の原因は、非線形計画問題には一般に大域的最適解ではない局所最適解が存在することである。解析的な手法を用いる限り局所最適解を回避することは難しいので、目的関数が凸関数の場合だけを対象とするのが普通である。

そうとは言え、目的関数が凸関数でなくても、局所最適解にとらわれることなく大域的最適解を求めたい。そのような場合には解析的な手法ではなく、確率的な手法が用いられる。本研究では、確率的な手法の一つでありニューラルネットワークの一種でもあるボルツマン・マシンを応用することで非線形計画問題の解法を提案する。

ボルツマン・マシンは、確率的な動作によりエネルギー関数と呼ばれる形式の関数の大域的最小解を求めることができる。しかしながらエネルギー関数は 2 値変数のベクトルの 2 次形式という非常に限定された形式であるため、そのままでは一般の非線形計画問題の目的関数の最小化には用いることができない。組み合わせ最適化問題に限れば、問題の目的関数の形がボルツマン・マシンのエネルギー関数の形と類似していて、それらを 1 対 1 で対応させることができるので、ボルツマン・マシンを使って問題を解くことができる。

しかし一般の非線形計画問題の目的関数の形には、とくに制限や法則性がないので、1 対 1 で対応するエネルギー関数を見つけることは不可能であろうと考えられる。そのため、これまではボルツマン・マシンを一般の非線形計画問題の解法として用いようとした研究は行われてこなかったようである。

そこで本研究ではボルツマン・マシンにある種の学習を取り入れることによって、エネルギー関数を目的関数にうまく対応させることを考えた。そして、学習によって試行錯誤的にエネルギー関数の形を変化させ、その最小値を求めることによって一般の非線形計画問題の大域的最適解を求めるための手法を開発した。

2. 非線形計画問題

非線形計画問題を定式化したものを下に示す。

$$\begin{array}{lll} \text{目的関数} & f(\boldsymbol{x}) \rightarrow \text{最小化} & \\ \text{制約条件} & g_i(\boldsymbol{x}) \le 0 \quad i=1,\ldots,m & (2.1) \\ & h_j(\boldsymbol{x}) = 0 \quad j=1,\ldots,l & \end{array}$$

目的関数 $f(\boldsymbol{x})$ は任意の形が許されるので、これを解く場合には、関数の形を制限したり、制約条件のつかない場合を考えたり、より簡単な形に分割して考えたりといったような措置を行うことが多い。

3. ボルツマン・マシン

ボルツマン・マシンは確率的な動作をする相互結合型のニューラルネットワークの一種である。ボルツマン・マシンでは、ニューラルネットワークを構成する各ニューロンを、伝統的な理由からユニットと呼ぶ。相互に結合している全てのユニットの出力の集合を"ネットワークの状態" と呼ぶことにすれば、各ユニットが非同期的に状態変化を繰り返していくことにより、ネットワークそのものの状態も変化していくことになる。このようなネットワークの動作を特徴付ける指標であり、ネットワークの平衡状態を調べるために導入されたのが、下に示すエネルギー関数である。

$$E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}z_iz_j+\sum_{i=1}^{n}\theta_iz_i \qquad (3.1)$$

ここで z_i は i 番目のユニットの出力（1 または 0）、w_{ij} はユニット i,j 間の結合係数、θ_i は i 番目のユニットのしきい値をあらわす。ただし、自己結合 w_{ii} は常に $w_{ii}=0$ である。

各ユニットは出力として 0 または 1 を取るが、どちらになるのかは決定論的に決まるのではなく 1 を出力する'確率' だけが決まる。

i 番目のユニットの出力 z_i が 1 となる確率を

$$p_i(z_i=1)=\frac{1}{1+\exp(-u_i/T)} \qquad (3.2)$$

$$u_i=\sum_j w_{ij}z_j+\theta_i \qquad (3.3)$$

と定義する。

ここで、T は温度と呼ばれる正のパラメータである。ボルツマン・マシンは、ある温度 T において、ネットワークの状態を式 (3.2) で与えられる確率 p_i にしたがって非同期的に更新するという操作を繰り返し行えば、十分に時間が経過した後には、ネットワークは確率的な平衡状態に達することになる。このような平衡状態においては、ネットワークの状態 α がエネルギー E_α をもつ確率 P_α は、統計物理学の分野でボルツマン分布と呼ばれる確率分布

$$P_\alpha=c\cdot\exp\left(\frac{-E_\alpha}{T}\right) \qquad (3.4)$$

に従うことが証明されている。ここで c は確率の総和を 1 にするための正規化定数であるので、P_α は状態 α のエネルギー E_α と温度 T のみに依存している。このように、平衡状態において確率分布がボルツマン分布になることが、ボルツマン・マシンと呼ばれる由来である。

このような平衡状態において、ネットワークの状態 β がエネルギー E_β をとる確率を P_β とすれば

$$\frac{P_\alpha}{P_\beta}=\frac{\exp(-E_\alpha/T)}{\exp(-E_\beta/T)}=\exp(-(E_\alpha-E_\beta)/T) \qquad (3.5)$$

なる関係式が得られる。このことより、$E_\alpha<E_\beta$ であれば、$\exp(-(E_\alpha-E_\beta)/T)>0$、$P_\alpha/P_\beta>1$ となり、$P_\alpha>P_\beta$ となる。したがって、ネットワークが平衡状態になれば、よりエネルギーの低い状態を取る確率が高くなることがわかる。したがって、もっともエネルギーの低い状態（すなわちエネルギー関数の大域的最小値）を取る確率はもっとも高くなる。

式 (3.1) で与えられる p_i と入力の重みつき総和 u_i の関係を、温度 T をパラメータとして示すと次の図のようになる。

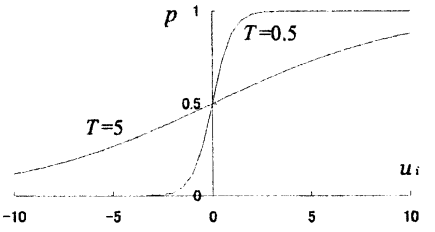


図 3.1 ボルツマン・マシンの遷移確率と温度の関係

ここで、この図からもわかるように、温度Tが低くなれば p_i の傾きは急になり、ネットワークの動作はより決定論的な動作に近くなる。特に $T \to 0$ の極限においては、p_i は階段関数になり、ホップフィールドモデルに帰着する。逆に T の値が大きくなればなるほど p_i の傾きはなだらかになり、u_i の値に依存しないでほぼ一定の確率1/2に近づくので、ボルツマン・マシンの各ユニットは、周りのユニットの影響を受けにくくなる。$T \to \infty$ の極限では、完全に確率的な動作を取る。このことは温度 T が高い場合、エネルギーがより高い状態であっても遷移する可能性が高まることを示している。そのようなエネルギーの高い状態への遷移は、ボルツマン・マシンが局所的な最小値に落ち込んでいる時、そこから抜け出すきっかけを与えることになる。そこで、はじめは比較的高い温度から開始し、平衡状態に到達してから、平衡状態を崩さないように徐々に温度を下げていくことが必要となる。そのために必要な温度制御に関する議論は、ここでは割愛する。

4. 提案する手法

4.1. アルゴリズム

本研究で提案するボルツマン・マシンは、一種の強化学習に相当する学習を行う。

強化学習（reinforcement learning）とは、あるシステムが試行錯誤を通じて環境に適応する学習制御のことである。正しい答えがわかっている場合の教師付き学習（supervised learning）とは異なり、報酬というスカラー値の情報のみを手がかりに学習を行う。

本研究で考えた方法は、ボルツマン・マシンというシステムが、試行錯誤を通じて、与えられた非線形計画問題という環境に適応するとみることもできるので、強化学習といえるのではないかと筆者は考えている。

強化学習には、システムを評価して報酬を与えるための評価基準が必要である。この評価基準として、非線形計画問題の目的関数を用いる。ボルツマン・マシンのユニットの状態をビット列と見なし、それを実数値に変換して目的関数の値を計算し、評価値として使用するのである。

あるユニットの出力の変化によって目的関数値が減少すれば正の報酬を与える。目的関数値が減少したということは、より望ましい状態に近づいたと考えることができるからである。

正の報酬とは、その状態でのエネルギー関数の値が減少するようにボルツマン・マシンにバイアスをかける事である。

つまり、再度そのユニットで同じ方向（0から1へ、もしくは1から0へ）の出力変化が起こりやすいように結合係数を調節する。

反対に目的関数値が増加すれば負の報酬を与える。

では、エネルギー関数を増減させるようなバイアスとはいったいなんであろうか。

それは、エネルギー関数の式(3.1)からも明らかなように、ユニット間の結合係数とユニット自身のしきい値を修正することである。

その修正ルールを簡単にまとめると、ユニットの出力の変化と、そのときの評価関数値の変化によって以下の表のようになる。

	ユニットの出力の変化	
評価関数値	0 → 1	1 → 0
減少	w_{ij} 増加	w_{ij} 減少
増加	w_{ij} 減少	w_{ij} 増加

表 4.1 強化学習のルール

また、制約条件のある問題において制約条件が満たされなかった場合には、評価関数が増加したものと見なして強化学習を行う。

なお、強化学習を行う際、結合係数の増減量は具体的にはどのような値が良いのか、現段階

では試行錯誤中であるが、今回は評価関数値の増減量に依存せず常に一定とする。ほかに増減量を決めるためのより良い方法があるかもしれない。

このような強化学習の方法は、いかにも近視眼的すぎるように感じられるかもしれない。直前の状態との比較だけで評価を決定してしまうからである。代替案としては、それまでで最良の評価を得た状態との比較から学習するといった方法も考えられるが、現段階では表 4.1 のルールを採用する。

なお、今回提案する手法では、結合係数の値に上限・下限を設けている。その上限を上回ったり下限を下回ったりするような学習は行われない。

本論文で提案するボルツマン・マシンは、はじめのうちは結合係数の調節のために動作しているので、はじめのうちは温度制御を行わない。あまり早いうちから温度を下げてしまうと、適切な結合係数が得られないうちに収束してしまうことになる。つまり、局所最適解におちいってしまう恐れがある。

そこで、十分な繰り返しののちに結合係数がほぼ確定したと思われる時点から温度制御を開始する。しかしながら、結合係数が確定したことを見極めるのが難しいので、現段階では一定回数以上繰り返した後に温度制御を開始することとする。

以上のことを踏まえて、本研究で提案する手法の簡単なアルゴリズムを以下に示す。

1. ボルツマン・マシンの各ユニットの初期出力、および各ユニット間の結合係数と各ユニットのしきい値をランダムな値で初期化する。
2. ボルツマン・マシンのユニットの中から一つをランダムに選択する。
3. 選択されたユニット i の出力 x_i が 1 となる確率を (3.2) 式によって決定する。
4. 3 で求めた確率に基づいてそのユニットの新しい出力値(1 か 0 か)を決定する。新しい出力値が前回の出力値と変化がなければ 2 へ戻る。
5. ユニットの出力値が変化すれば、ボルツマン・マシンのユニットの状態も変化しているはずである。そこで、その状態に対応する評価関数の値を計算する。
   その値を前回の評価関数の値と比較する。
   値が変化していなければ、2 へ戻る。
6. 選択されたユニットの結合係数としきい値を、評価関数値の増減に基づいて変更する。(強化学習)
7. ある一定の回数以上繰り返したあとであれば、以下の式にしたがって温度制御を行う。

   $$T(t)=\frac{T_0}{\log(1+t)}$$

   この式は、参考文献[3]から来ている。
   ここで t は温度制御を開始してからのステップ数である。
8. 2 へ戻る。

このアルゴリズムの終了条件は、ユニットの状態遷移が収束したと見なされる場合か、指定の回数の繰り返しを行った場合とする。

4.2. 動作原理

学習を取り入れたボルツマン・マシンの動作原理を、簡単な例をもちいて説明する。

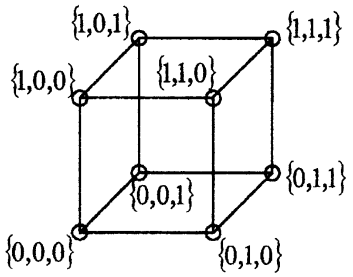
まず、3 つのユニットからなるボルツマン・

マシンを考える。

このボルツマン・マシンは、8通りの状態を取ることができる。

すなわち、各ユニットの出力z_iを用いてそれらの状態を$\{z_1,z_2,z_3\}$と表記すると、$\{0,0,0\}$, $\{0,0,1\}$, $\{0,1,0\}$, $\{0,1,1\}$, $\{1,0,0\}$, $\{1,0,1\}$, $\{1,1,0\}$, $\{1,1,1\}$ の8通りである。

これら8通りの状態を、立方体の頂点に対応させてみると、下の図のようになる。

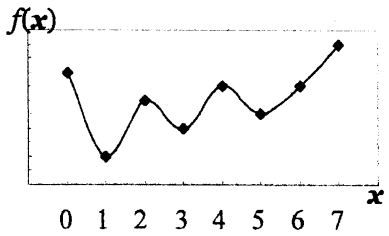


立方体の各頂点はボルツマン・マシンの状態であり、辺で結ばれた頂点同士は、互いに1つのユニットの変化だけで遷移することができる状態である。

この例ではユニット数が3なので3次元で表現できるが、一般にユニット数が n 個の場合は n 次元超立方体を考えれば、ボルツマン・マシンの状態をその超立方体の各頂点と対応付けられる。

ここから、この3ユニットのボルツマン・マシンを使って、前節で述べたアルゴリズムを実行してみる。

例として解いてみる問題は以下のグラフで表されるような目的関数を持つものとする。

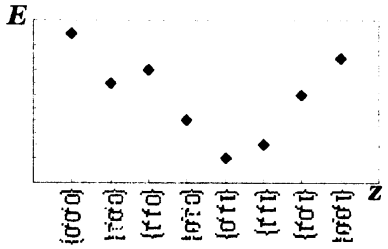


この目的関数$f(x)$の大域的最小値は$x=1$の点である。目的関数の具体的な値はここではさほど重要ではないのでとくに示さない。xの値によって$f(x)$の値が上下しているということだけ読み取れればよい。

まず、アルゴリズムの手順1として、各ユニットの出力、各ユニット間の結合係数、各ユニットのしきい値をランダムに決定する。

今回は、初期状態が$\{1,1,1\}$になったとする。簡単のため、結合係数としきい値も具体的な値は考えないが、結合係数としきい値が決まると、エネルギー関数を計算することができる。

そこで仮に各状態におけるエネルギー関数の値を計算したこととして、それをグラフにプロットしてみることにする。グラフの横軸にはボルツマン・マシンの各状態をとる。ここでのポイントは、各状態をグレイコードとみなして並べることである。縦軸にはエネルギー関数の値をとる。



結局、エネルギー関数が上記のグラフのようになったとする。エネルギー関数の値自体はさほど重要ではないので、ここでは具体的な値はあえて示さない。

もし仮に、最初に決めたランダムな結合係数としきい値を変更せずにこのままボルツマン・マシンを動作させると、最終的に$\{1,1,0\}$の状態にたどりつくはずである(正確には、$\{1,1,0\}$の状態になる確率が最も高くなる)。なぜならば、その状態のエネルギー関数値がもっとも低いからである。

ここで、アルゴリズムの手順2に移る。

ランダムにユニットをひとつ選ぶ。今回は

z_3 を選ぶことにする。

アルゴリズムの手順3、4では、選んだユニットの出力が変化する確率を決定し、その確率に基づいて出力値を決定する。今回は z_3 が 1 から 0 になることとする。

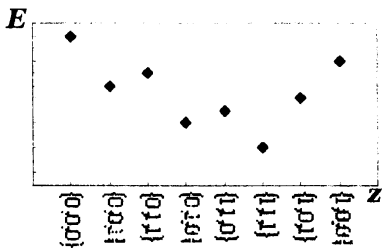
手順5では、評価関数すなわち問題の目的関数の値を計算する。z_3 が 0 になったので、ボルツマン・マシンの状態は $\{1,1,0\}$ になっている。この状態を、グレイコードによって数値に変換すると、"4" に相当する。$x=4$ の場合の目的関数 $f(x)$ の値は、状態変化前の値よりも増えている。

そこで、手順6の学習に入る。

学習フェーズでは、この場合は負の報酬が与えられる。負の報酬を与えるということは、ボルツマン・マシンはその状態変化が起こりにくくなるように学習するということである。つまり、結合係数などを調節して、$\{1,1,0\}$ の状態に対応するエネルギー関数の値を増加させるのである。そうなることで、最終的には $\{1,1,0\}$ の状態になる確率が低下する。

手順7の温度制御は、まだ学習中なのでここではまだ行わない。

これでイテレーションが1回終わったが、この時点でエネルギー関数のグラフが下記のグラフのようになったとする。



$\{1,1,0\}$ に対応するエネルギー関数の値が変化している。(本当ならば、ほかの状態に対応するエネルギー関数値も変化するはずであるが、ここでは説明の簡略化のために敢えて変化させなかった)

もし仮にこのままボルツマン・マシンを動作させると、さきほどとはかわって最終的には $\{1,1,1\}$ の状態にたどりつくはずである。

次は手順2に戻って、ランダムにユニットを選ぶ。次は z_1 を選択することにしよう。

手順3、4によって z_1 の出力の変化を決定する。ここでは、1から0に変化することとする。

手順5では評価関数の値を計算する。z_1 が 0 になったのでボルツマン・マシンの状態は $\{0,1,0\}$ となっている。この状態をグレイコードと見なして数値に変換すると $x=3$ の状態に相当する。その場合の $f(x)$ の値は、グラフから読み取ると今度は減少している。

手順 6 ではその評価関数値に対する学習を行う。ここでは学習の結果、状態 $\{0,1,0\}$ に対するエネルギー関数の値が減少するように結合係数などの修正が行われる。

以降、同様にイテレーションを行っていくと、途中で局所最適解に対応する状態のエネルギー関数値が最も低くなってしまうこともあるが、十分な学習フェーズの繰り返しの後には、大域的最適解に対応するエネルギー関数の値が最も低くなるはずである。大域的最適解に相当する状態に遷移した場合は、必ず正の報酬が与えられるからである。また、直接大域的最適解の状態に遷移しなくても、近傍の状態への遷移が行われた時点で行われる学習により、間接的に大域的最適解の状態のエネルギー関数の値も低下することがわかっている。

さて、大域的最適解に対応する状態のエネルギー関数の値が最も低くなった時点で、温度制御を開始することができるようになる。温度制御を行いながら、ボルツマン・マシンを動作させると、エネルギー関数が最低となる状態に到

達して収束する。収束するとは、最適状態の出現確率が 1 となることである。

そうなることで、このボルツマン・マシンによってこの問題を解くことができたといえることになる。

以上は 3 つのユニットからなるボルツマン・マシンを用いた説明であったが、ユニット数を増やせば状態数も増え、目的関数の計算に使用する変数値の範囲も広げることができるということがわかるだろう。

また上記の例では、状態をビットパターンと見なし、グレイコードを用いて整数への変換を行ったが、同様に固定小数点数に変換することも容易である。浮動小数点数への変換は実験では行ったことはないが、もし行ってみれば興味深い結果が得られるかもしれない。

5. 実験

ここでは、前節で提案したアルゴリズムで実際に非線形計画問題をとくことができるのかどうかを実証するべく、いくつかの実験を行う。

はじめに、最も簡単な非線形計画問題として、1 変数、制約条件なしという問題を解いてみる。

その問題の目的関数に局所的最適解が存在しても、大域的最適解に収束するかどうかを調べる。

次に、変数を一つ増やして 2 変数の問題としてみる。この場合も、制約条件がない問題を対象とする。また、非凸関数であることはもちろんのこと、微分不可能な関数についても大域的最小値を探索できるかどうかを調べる。前節で提案したアルゴリズムには、本来は目的関数の微分可能性は関係ないが、解析的手法に対する優位性を確認するために実験を行う。さらにその後、簡単な制約条件をつけた問題や多変数の問題を解いてみる。そして最後に、非線形計画問題を解くアルゴリズムの性能評価に使われるベンチマーク関数の大域的最小値を求めてみる。

なお、実験では多量の乱数を用いるが、乱数の発生には Mersenne Twister (MT)という疑似乱数発生アルゴリズムをもちいた[4]。この MT は、第 24 番目のメルセンヌ素数 $2^{19937}-1$ というきわめて長い周期と、623 次元超立方体の中に均等に分布するという優れた特徴をもつ。MT はモンテカルロ法をはじめ、数値シミュレーションで近年特に用いられている優れた乱数発生アルゴリズムの一つである。

5.1. 1 変数の問題

最初に着手した問題は、次のようなものである：

Minimize $\quad f(x)=\cos 3x+x^2-x \quad (5.1)$

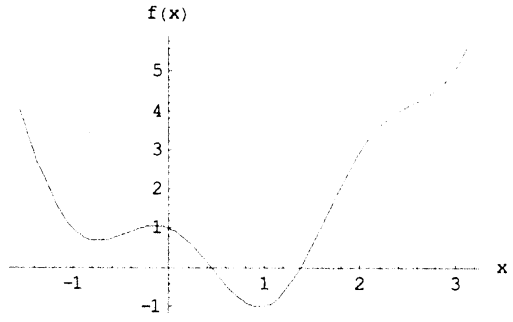


図 5.1 式 (5.1) のグラフ

この関数には $x=0.946$ に大域的最小値が存在し、$x=-0.946$ の点に局所的最小値が存在する。

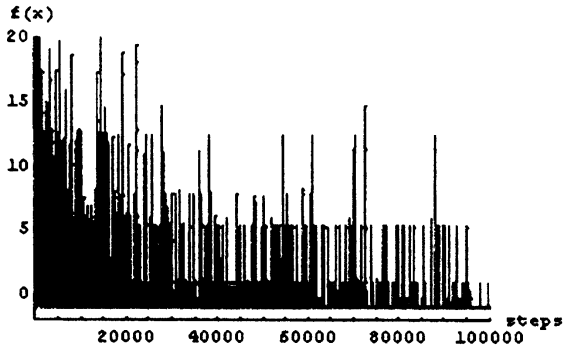
この問題を解くためのボルツマン・マシンとして、ユニット数 16 のボルツマン・マシンを用意した。

すなわち、16 ビットの状態数が存在するので $2^{16}=65536$ とおりの状態数が扱えることになる。

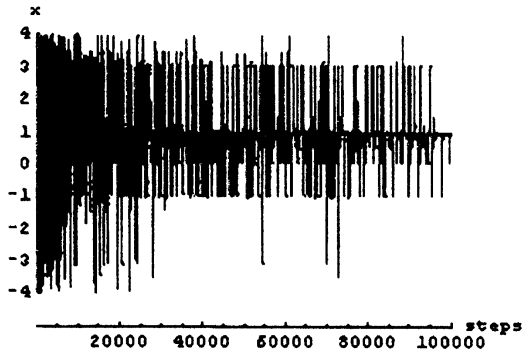
今回はこの 16 ビットを固定小数点数と見なして、そのうち 13 ビットを小数部、残りの 3 ビットを整数部とした。

また、ビットパターンを固定小数点数と見なす際に、普通の 2 進数ではなく、グレイコードを用いた。それによって、1 分解能分の変化は常に 1 ビットの変化となる。

はじめに、初期値を $x=-1$、温度 T を $T=1/\log 2=1.442695$ で一定に保って（つまり温度制御を行わずに）、10 万ステップ繰り返した場合の様子を見てみる：



目的関数値の収束状況

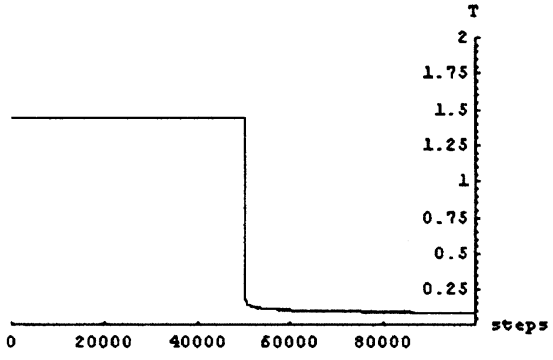


変数 x の値の収束状況

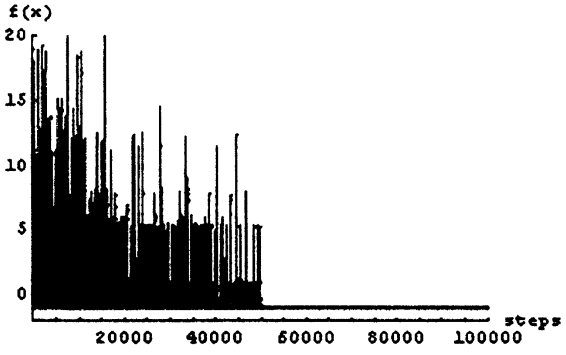
目的関数 $f(x)$ の値はステップを繰り返すにつれておよそ -1 に、変数 x の値はおよそ 0.947 に収束していきつつある。しかし、まれに目的関数値が大きくなるような状態遷移も起こっている。（グラフでは頻繁に状態遷移が起こっているように見えるが、このグラフの横軸が 10 万ステップにもおよぶために凝集してしまっているのでそう見える。実際の頻度はそれほど高くない。）

このままボルツマン・マシンを動作させつづけても、確率分布的な平衡状態に達してはいても一つの値に収束するということはない。そこで、温度制御を行う必要が出てくる。

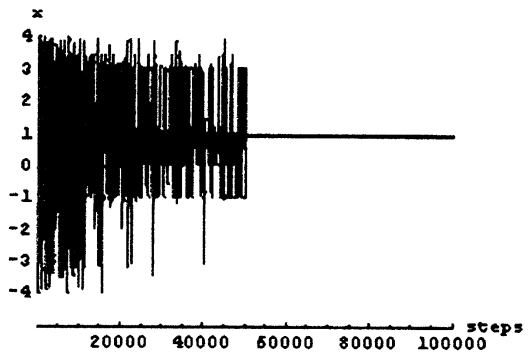
次は、5 万ステップまでは結合係数の強化学習のために温度制御をせずに $T=1/\log 2=1.442695$ に保ち、5 万ステップを過ぎたところから $T=1/\log(t+2)$（ただし t はステップ数−50000）というスケジュールにしたがって温度制御を行った。x の初期値は先ほどとおなじ $x=-1$ である。



温度 T の推移



目的関数値の収束状況



変数 x の値の収束状況

グラフを見ると、5 万ステップ繰り返して温度制御を開始した直後に収束したことがわかる。

収束した x の値は $x=0.946411$ であり、そのときの目的関数の値は $f(x)=-1.005354$ となった。この値は 16 ビットの固定小数点数で求められる大域的最適解である。

5.2. 2 変数の問題

次は変数が 1 つ増えて 2 変数となっても、提案するアルゴリズムが有効であるかどうかを見ていく。

5.2.1. 微分可能な目的関数の場合

まず、微分可能な関数を考える。

取り上げる例題を以下に示す：

$$\text{Minimize} \quad f(x,y)=x^2\left(4-2.1x^2+\frac{1}{3}x^4\right)+xy+y^2\left(-4+4y^2\right) \qquad \cdots(5.2)$$

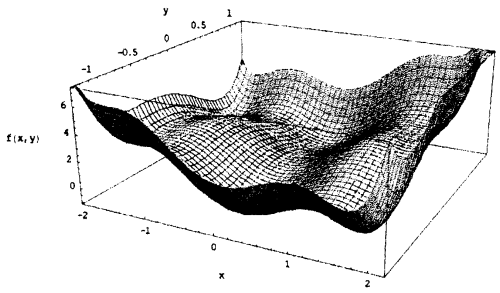


図 5.2 式 (5.2) のグラフ

この関数には大域的最小値が 2 ヶ所あり、その座標は $(0.09,-0.71)$ と $(-0.09,0.71)$ である。

また、そのほかに局所的最小値が 4 ヶ所存在する。

このグラフの等高線は以下の図のようになる：

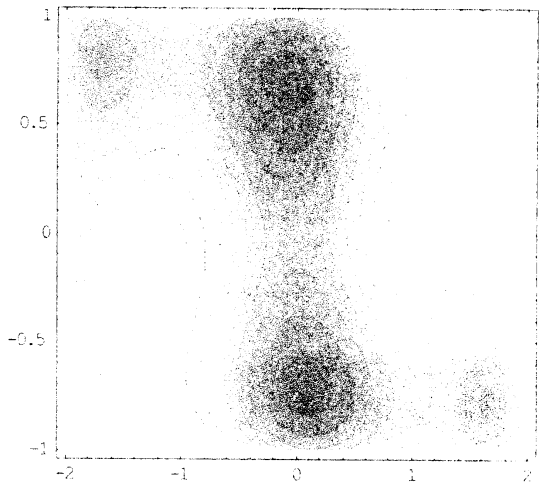


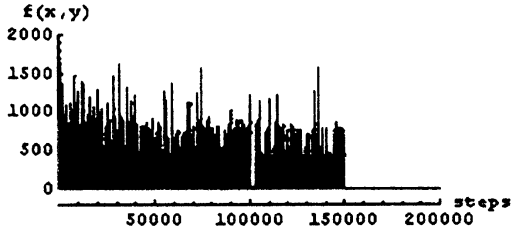
図 5.3 式 (5.2) の等高線

図中の色が濃くなっているところがより目的関数 $f(x,y)$ の値が小さくなっているところである。

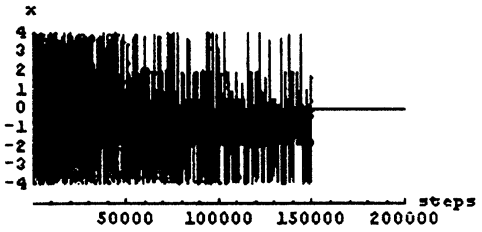
この問題を解くためのボルツマン・マシンとして、ユニット数 32 のボルツマン・マシンを用意した。

変数 x と y のためにそれぞれ 16 ビットずつ用い、それぞれ固定小数点数とする。小数部のビット幅は 5.1 節と同様に 13 ビットとする。グレイコードを用いるのも同様である。

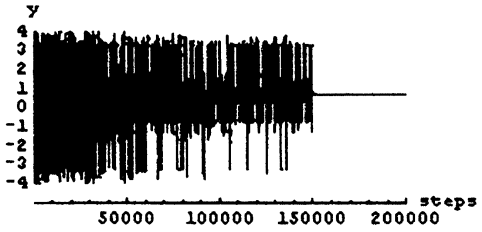
今度は 15 万ステップまで結合係数の学習のために温度を $T=1/\log 2=1.442695$ で一定に保ち、15 万ステップを過ぎたところから $T=1/\log(t+2)$（ただし t はステップ数－150,000）というスケジュールにしたがって温度制御を行った。



目的関数値の収束状況



変数 x の収束状況



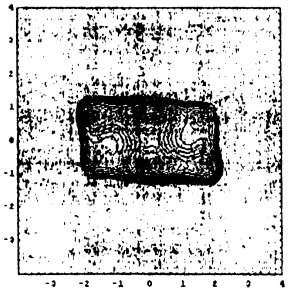
変数 y の収束状況

今度は 1 つの値に収束した。

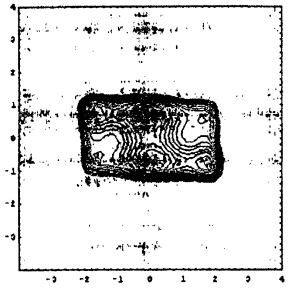
収束した x の値は $x=$-0.089600、y の値は $y=0.713013$ であり、そのときの目的関数の値は $f(x,y)=$-1.031627 となった。この値は 16 ビットの固定小数点数で求められる大域的最適解である。

ボルツマン・マシンが解を探索して収束していく様子を見るために、20 万ステップの探索点を 4 段階に分けて、等高線の上にプロットした。

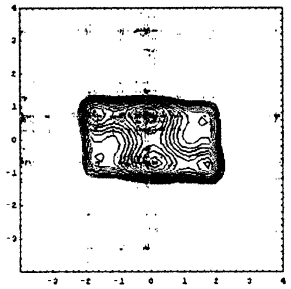
ステップ 1 ～ 50000



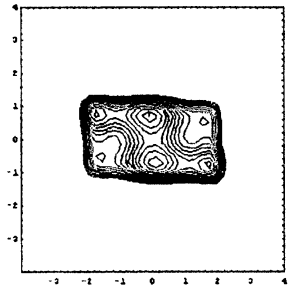
ステップ 50000～100000



ステップ 100000～150000



ステップ 150000～200000



最初のうちは探索点が広く分布しているが、ステップ数が進むにつれて、温度制御を行わなくても少しずつ収束しているのがわかる。

また、今回は大域的最適解のうちの一つ、$(x,y)=(-0.09,0.71)$ の方に収束したが、もう一方の解に収束する場合もある。どちらの大域的最小解に収束するかはそのとき次第であり、初期値などの値には依存しないようである。

5.2.2. 微分不可能な目的関数の場合

次は微分不可能な問題を考える。

取り上げる例題を以下に示す：

Maximize $f(x,y) = \mathrm{Max}[-0.5(x-1)^2 - 4.5(y+2)^2 + 33,$
$30 - 1.25y^2 - 2.5(x+1)^2(y-1)^2,$
$-0.25(x+4)^2 - 0.8(y-5)^2 + 5,$
$-35(x+3)^2 - 52(x+3)(y-4.3) - 65(y-4.3)^2 + 34,$
$-0.05(x+1)^2(y+1)^2 - 0.75y^4 + y^3 + 9y^2 - 10]$

$\cdots(5.3)$

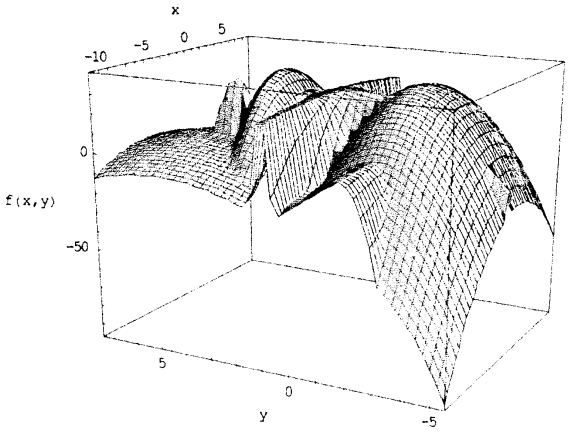


図 5.4 式 (5.3) のグラフ

この関数は、5 本の式を組み合わせて、それらのうちの最大となる部分をつなぎ合わせていったような形になる。

式 (5.3) の問題は最小化ではなくて最大化問題である。本論文で提案するアルゴリズムは最小化問題しか解くことができないが、このような場合は式 (5.3) の全体に -1 をかけて最小化問題に変換すればよい。

Minimize $f(x,y) = -\mathrm{Max}[-0.5(x-1)^2 - 4.5(y+2)^2 + 33,$
$30 - 1.25y^2 - 2.5(x+1)^2(y-1)^2,$
$-0.25(x+4)^2 - 0.8(y-5)^2 + 5,$
$-35(x+3)^2 - 52(x+3)(y-4.3) - 65(y-4.3)^2 + 34,$
$-0.05(x+1)^2(y+1)^2 - 0.75y^4 + y^3 + 9y^2 - 10]$

$\cdots(5.4)$

式 (5.4) の形は、式 (5.3) の形をひっくり返したものとなる。

この関数には大域的最小解が 1 か所存在して、その座標は $(-1.0, 3.0)$ である。また、そのほかに局所的最小値が 4 か所存在する。

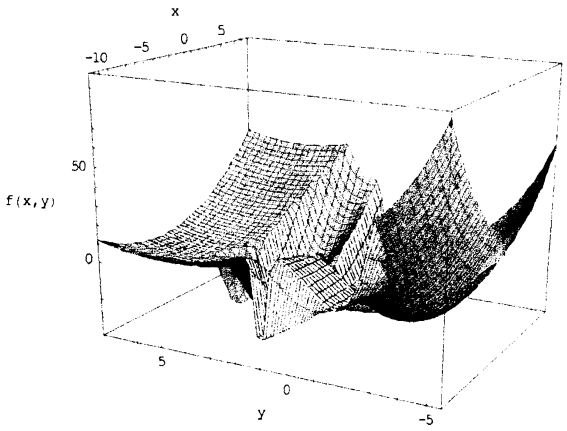


図 5.5 式 (5.4) のグラフ

式 (5.4) の等高線は以下のようになる：

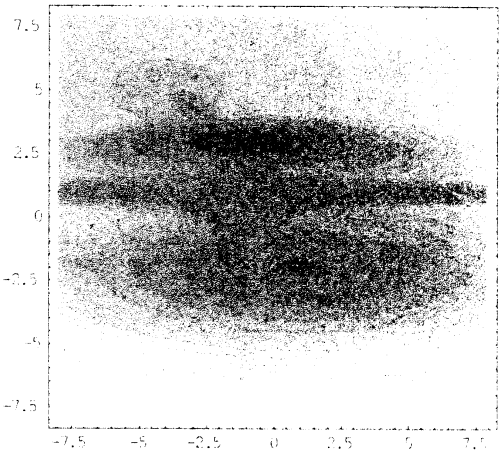


図 5.6 式 (5.4) の等高線

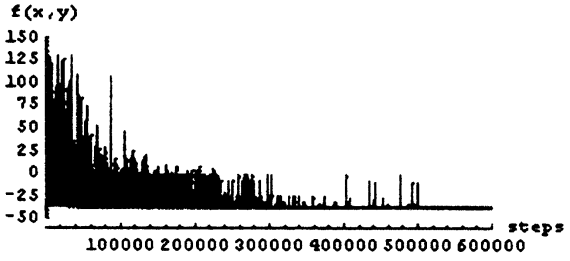
図中の色が濃くなっているところが、より目的関数 $f(x,y)$ の値が小さくなっているところである。

この問題を解くためのボルツマン・マシンとして、ユニット数 32 のボルツマン・マシンを用意した。

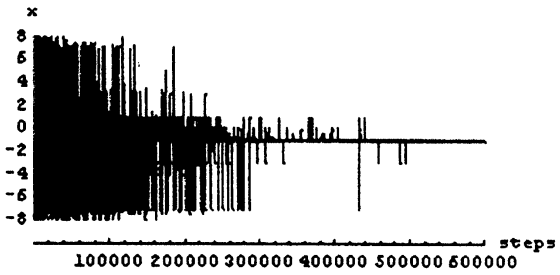
変数 x と y のためにそれぞれ 16 ビットずつ用い、それぞれ固定小数点数とする。小数部のビット幅は今回は 12 ビットとした。グレイコードを用いたのは同様である。

50 万ステップまでは、結合係数の強化学習

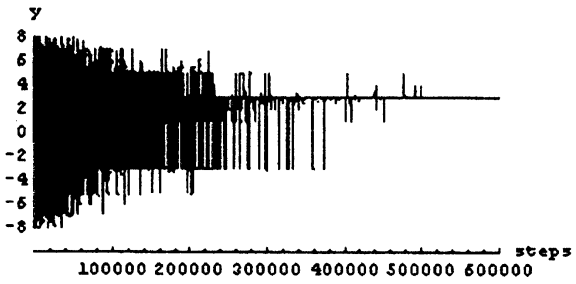
のために温度制御をせずに温度 T を $T=1/\log 2=1.442695$ で一定に保ち、50 万ステップを過ぎたところから $T=1/\log(t+2)$ (ただし t はステップ数−500,000) というスケジュールにしたがって温度制御を行った。初期値は先ほどと同じ $(x,y)=(5.0,-5.0)$ である。



目的関数値の収束状況



変数 x の収束状況



変数 y の収束状況

収束した x の値は $x=$-1.000000、y の値は $y=3.000000$ であり、そのときの目的関数 $f(x,y)$ は $f(x,y)=$-37.25000 となった。

このことから、5.2.1 の場合と比較して、同じ 2 変数の問題であっても目的関数の形によっては結合係数の学習に要する時間が異なることがあるということがわかる。

5.2.3. 制約条件のついた問題

次は制約条件のある問題を考える。

取り上げる例題を以下に示す：

$$
\begin{aligned}
\text{Minimize} \quad & f(x,y)=-\mathrm{Max}\,[-0.5(x-1)^2-4.5(y+2)^2+33,\\
& 30-1.25y^2-2.5(x+1)^2(y-1)^2,\\
& -0.25(x+4)^2-0.8(y-5)^2+5,\\
& -35(x+3)^2-52(x+3)(y-4.3)-65(y-4.3)^2+34,\\
& -0.05(x+1)^2(y+1)^2-0.75y^4+y^3+9y^2-10],\\
\text{Subject to} \quad & y-x\le 0.
\end{aligned}
\qquad \cdots(5.5)
$$

この例題は、5.2.2 節の式 (5.4) に簡単な制約条件 $y-x\le 0$ がついただけのものである。下の図で、グレーで塗りつぶされた範囲が探索範囲から除かれる。

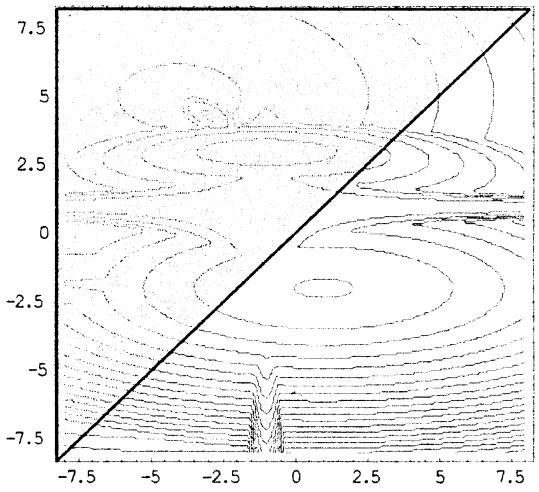
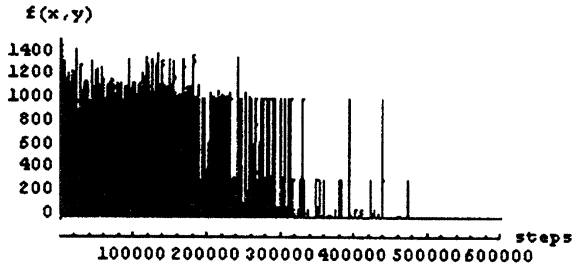


図 5.7 制約条件付きの問題 (5.5) の等高線と制約範囲

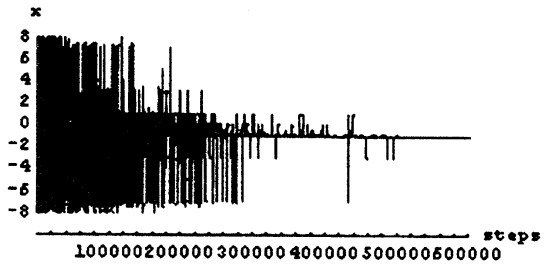
この問題を解くためのボルツマン・マシンとして、ユニット数 32 のボルツマン・マシンを用意した。

変数 x と y のためにそれぞれ 16 ビットずつ用い、それぞれ固定小数点数とする。小数部のビット幅は 5.2.2 節と同様、12 ビットとした。グレイコードを用いたのも同様である。

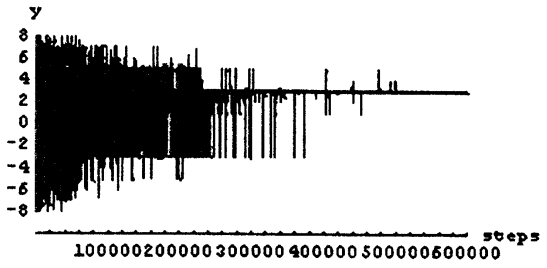
この問題を解く際の温度制御は、5.2.2 節と同様に 50 万ステップまでは $T=1/\log 2=1.442695$ で一定、50 万ステップを過ぎたところから $T=1/\log(t+2)$（ただし t はステップ数−500,000）というスケジュールで行った。初期値も先ほどと同じ $(x,y)=(5.0,-5.0)$ である。



目的関数値の収束状況



変数 x の収束状況



変数 y の収束状況

最終的に、$x=1.000000$、$y=-2.000000$、目的関数値 $f(x,y)=-33.000000$ に収束した。

5.3. ベンチマーク関数

本研究で提案した手法が、果たして実用にたえるだけの性能を持つか、あるいはすくなくともその可能性があるかどうかを調べるために、一般の非線形計画問題の解法に対すベンチマーク関数としてよく使われる関数をとりあげ、その最小値を求めてみた。

ここで取り上げるのは、

(1) Rastrigin 関数
(2) Schwefel 関数
(3) Rosenblock 関数

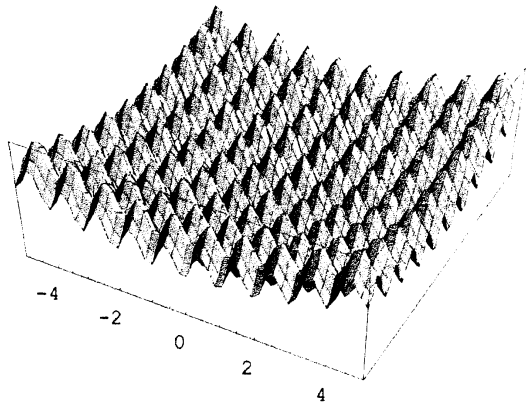
の 3 種類である。

5.3.1. Rastrigin 関数

Rastrigin 関数は、以下の式で表される。

$$f_{rastrigin}=n\times 10+\sum_{i=1}^{n}\left(x_i^{\,2}-10\cos(2\pi x_i)\right) \quad \cdots(5.6)$$

2 次元 $(n=2)$ のときのグラフを以下に示す。



グラフからもわかるが、局所的な最小値が多数存在している。一般的に、単純な遺伝的アルゴリズムが苦手とするといわれる多峰性の関数であるが、今回提案した手法では解くことができた。

この関数の大域的最小値となる点は、関数の次元によらず $x_i=0\ (i=1...n)$ の点である。

この関数を目的関数とする非線形計画問題を解いてみたところ、2 次元のときはもちろん、3 次元、4 次元と次元を増やしても大域的最適解を求めることができた。実験では 10 次元の

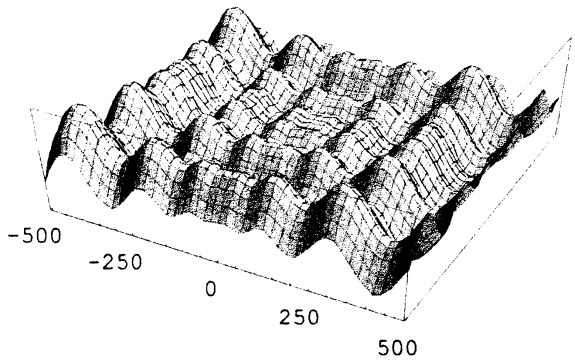
場合まで解いてみて、いずれも大域的最適解に到達することを確認した。
（11 次元以上はまだ実験を行っていない）

5.3.2. Schwefel 関数

Schwefel 関数は、以下の式で表される。

$$f_{schwefel} = \sum_{i=1}^{n}\left(-x_i \sin\left(\sqrt{|x_i|}\right)\right) \qquad (5.7)$$

2 次元$(n=2)$のときのグラフを以下に示す。



この関数も見てわかるとおり多峰性の関数である。

この関数は、外側に行けば行くほど値が小さくなるという形をしている。そのため、探索する範囲によって大域的最適値が変わってしまう。今回は $-512 \le x,y \le 512$ の範囲に限定して探索を行った。その場合の大域的最小値は $f(\boldsymbol{x}) = -n \cdot 418.9829$ 、 $x_i = 420.9687$ $(i=1...n)$ である

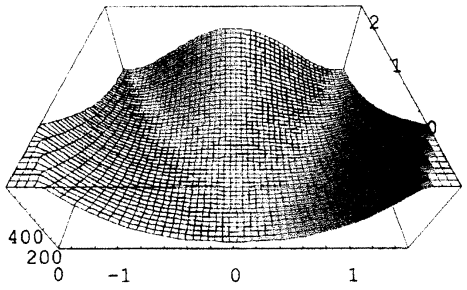
この関数を目的関数とする非線形計画問題を解いてみたところ、10 次元の場合までいずれも大域的最適解に到達することを確認した。

5.3.3. Rosenblock 関数

Rosenblock 関数は、以下の式で表される。

$$f_{rosenbrock} = \sum_{i=1}^{n-1}\left(100\left(x_i^{\,2} - x_{i+1}\right)^2 + \left(1-x_i\right)^2\right) \qquad \cdots(5.8)$$

2 次元$(n=2)$のときのグラフを以下に示す。



この関数は制約なしのベンチマーク関数の中では悪名高いもので、バナナ関数という別名でも知られている。等高線が U 字型の急傾斜となっているためそのような呼び名がついている。

この関数の最小値は $f(\boldsymbol{x}) = 0$ であり、そのときの座標は $x_i = 1$ $(i=1...n)$ となる。

この関数を目的関数とする非線形計画問題を解いてみたところ、大域的最小値のすぐ近くまでは到達することができるがそのあとはなかなか大域的最小値に近づかなくなってしまうという現象が確認された。すぐ近くというのは、16 ビット固定小数点（小数部 13 ビット）において、大域的最適解の点からの差が 1×10^{-3} 程度の近傍である。

本研究で提案する手法は、このバナナ関数のような形を苦手とするのかもしれない。

今後、原因の究明とその対策の必要がある。

6. 考察

6.1. この手法を適用できる問題の条件

本研究で提案したアルゴリズムは、例題で見た限りでは、温度制御を正しいスケジュールで行えば、多くの非線形計画問題の大域的最適解を求めることができるといえる。

そこで、温度制御を正しいスケジュールで行うことができるという前提で、本アルゴリズムは例題以外の問題ではどのような問題には適用できて、どのような問題には適用できないの

かを考察してみる。

(1) 目的関数が微分可能でない問題

目的関数に微分不可能な点が含まれていても、例題でも見たとおり、解くことができた。

本アルゴリズムでは微分を使わないので、目的関数の微分可能性は必要ないであろう。

したがって、たとえ大域的最適解になる点が微分不可能な点であったとしても、その点を見つけ出すことができると考えられる。

(2) 目的関数が連続でない問題

目的関数に不連続点が含まれていても、本アルゴリズムでは大域的最適解を見つけ出すことができると考えられる。本アルゴリズムはもともと離散的な手法なので、目的関数の一部が不連続であったとしても、その影響はないと見なすことができるであろう。

(3) 大域的最適解が複数・もしくは無数にある問題

大域的最適解が複数ある関数は、例題の5.2.1 でも見たとおり、そのうちのどれか一つにしか収束しない。どれに収束するかは完全に確率的であり、コントロールすることはできないだろう。その原因は、本アルゴリズムでは、ボルツマン・マシンの状態遷移確率分布のピークを大域的最適解に重なるように強化学習が行われているということである。状態遷移確率の分布はボルツマン分布であるので、ピークは一つしかできない。したがって大域的最適解も一つしか求められないということになる。

大域的最適解が複数・もしくは無数にある問題の場合については、今後さらなる研究の必要がありそうである。

(5) 整数計画問題

本アルゴリズムは離散的な手法であるといえるので、一般に難しいといわれている整数計画問題への応用も簡単にできるであろう。たとえば、16 ビットの固定小数点数で小数部のビット幅を 0 ビットにしてしまえば、16 ビット整数の変数による問題を解くことができると考えられる。

(6) 組み合わせ最適化問題

ボルツマン・マシンはもともと組み合わせ最適化問題を解くことができるので、本アルゴリズムでも解くことができるだろう。小数部のビット幅を 0 とした 1 ビット固定小数点数の変数を考えればよい。

従来の方法と違うのは、目的関数だけ設定すれば結合係数を算出しなくてもよいという点である。従来の手法では、目的関数とエネルギー関数を比較して結合係数の値を算出する必要があった。本アルゴリズムを適用する際は、結合係数の値をはじめはランダムにしておいてよい。

本アルゴリズムを適用することによって、強化学習で結合係数を適切な値へと設定できたとして、その結合係数と、従来の手法で式の形の対応から得られる結合係数とを見比べると、興味深い結果が得られるかもしれない。

7. まとめ

ボルツマン・マシンを使って非線形計画問題を解くという研究は従来なされてこなかったようであるが、その理由の一つとして、エネルギー関数の形式に制約があるということがあげられるだろう。

本研究では、ボルツマン・マシンに強化学習を取り入れたことにより、ボルツマン・マシン

のエネルギー関数の大域的最小点と、非線形計画問題の目的関数の大域的最小点を動的に一致させることができるようになり、結果としてその制約を克服することができた。

本稿では、そのボルツマン・マシンを使ったアルゴリズムを提案して、そのアルゴリズムを使うことによっていくつかの種類の非線形計画問題の大域的最適解を求めることができるということを示した。

温度制御の難しさや、大域的最小解が複数ある場合の問題点など、これから研究されるべき点はいくつかあるが、本アルゴリズムを実際的な問題の解法として応用することは十分に可能なのではないかと思う。

また、本アルゴリズムは、組み合わせ最適化問題、整数計画問題、凸計画問題、線形計画問題、2次計画問題などの一般に非線形計画問題の範疇に含まれる問題はすべて解くことができると考えられる。したがって、本アルゴリズムは非線形計画問題を解くための汎用的な手法の一つに仲間入りできるといえるのではないだろうか。

8. 参考文献

[1] Hinton,G.E., and T.J.Sejnowski, Learning and relearning in Boltzmann machines, in D.E.Rumelhart, J.L.McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol.1: Foundations, The MIT Press, Cambridge, pp. 282-317, 1986.

[2] 上坂吉則, *ニューロコンピューティングの数学的基礎*, 近代科学社, 1993.

[3] Geman,S., and D.Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.Pami-6 No.6, 721-741, 1984.

[4] Matsumoto,M., and T.Nishimura, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, *ACM Trans. on Modeling and Computer Simulation* Vol. 8, No. 1, January pp.3-30, 1998.